CS 7643 Fall 2024 Final Project: Cross-Attention Fusion of Multimodal Transformers for Forecasting Stock Price

GitHub Repository

Kai Alcayde Christopher Pak
Luan Lam Jake Yang

Abstract

Our project uses a multimodal transformer fusion approach via gated cross-attention to experiment with predicting the movements of stock prices, ideally bolstering predictions made from the individual composing Transformer-based models, which individually focus on different aspects such as time, vision, or positional encoding. Accurately predicting stock movements is a difficult problem that many wish to solve, in order to facilitate their investment decisions, get rich, and quit their 9-to-5 jobs. By implementing and fusing several transformer-based approaches we aimed to create a model that is able to predict stock movements with high accuracy - or at least gain some insight into similar approaches that might. With this in mind, our multimodal model was able to achieve a 75% total reduction of MSE from the minimum individual model MSE, with the majority of performance and accuracy improvement coming from the first week of our stock price prediction window.

1. Introduction

We attempted to implement three transformer-based models: a Temporal Fusion Transformer (TFT) [4], an Autoformer [6], and a Vision Transformer (ViT) [3]. We then attempted to use these models, along with a Large Language Model (LLM) Text Embedder , to optimize stock price-prediction. Utilizing 3 distinct data modalities in the forms of tabular data (TFT, Autoformer), visual data (ViT), and text data (LLM), our plan was to ultimately fuse their learned feature representations to get better stock price prediction results. We took inspiration from the paper "Stock Movement Prediction with Multimodal Stable Fusion via Gated Cross-Attention Mechanism" [8] to implement fusion. We optimized results through experimentation with stocks of contrasting properties (e.g high vs. low volatility), and with time series of different intervals (e.g day vs. intraday), to predict stock prices in varying windows of time. We varied the prediction date windows, and focused some models on producing results in the short-term date ranges, and others in long term date ranges to create a more robust model.

Today, stock price prediction is done through many different ways, ranging from Time Series Analysis, Supervised Learning, to Deep Learning and Transformers. These include some methods that we have explored in this project. An important thing to note about the methods of today, however, are that they tend to solely focus on these singular models, whether it be focusing on a single transformer model such as TFT or through Sentiment Analysis. This is where their limits are clear - they are limited to a single learned feature representation present in the training of its model, which are learned in restricted intervals - either short or long term.

If we are successful, we can revolutionize stock price prediction. By combining the learned feature representations of multiple models and varying our focus on prediction window lengths, we ideally improve the robustness of our overall predictions, compared to predictions made by singular, simple models.

For the transformer models, the data we used was from Tim Mooney's 'Stock Market Data' on Kaggle, specifically the S&P 500 Data [5]. It contained Date, Volume, High, Low, Open, Closing, and Adjusted Closing Price data for 403 of the world's leading companies. We also obtained static, unchanging stock data such as Debt to Equity Ratio for each company from Google. For creating text embeddings with the LLM, we used data from Hanna Yukhymenko's 'Stock Tweets for Sentiment Analysis and Prediction' on Kaggle [7], which contained 80744 tweets between 9/30/2021 to 9/30/2022 concerning the top 25 most watched stocks on Yahoo Finance. We restricted our training data window to dates from 1/1/2017 to 12/31/2021.

2. Approach

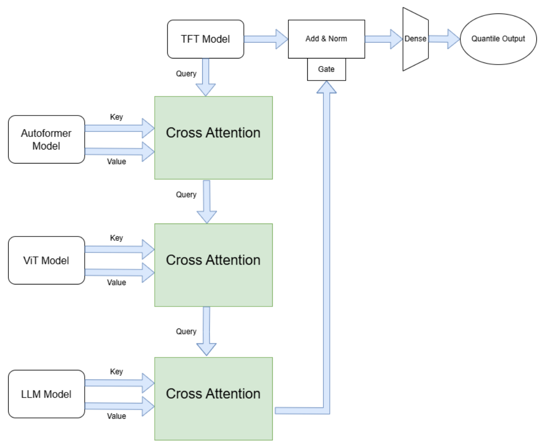


Figure 1. Graph of Fusion Model

$$MultiModalAttentionHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \widetilde{\mathbf{H}}\mathbf{W_H} \quad (1)$$

$$\begin{aligned}\widetilde{\mathbf{H}} &= \widetilde{A}(\mathbf{Q}, \mathbf{K})\mathbf{V}\mathbf{W}_V, \\ &= \left\{\frac{1}{H}\sum_{h=1}^{m_H} A(\mathbf{Q}\mathbf{W}_Q^{(h)}, \mathbf{K}\mathbf{W}_K^{(h)})\right\}\mathbf{V}\mathbf{W}_V, \\ &= \frac{1}{H}\sum_{h=1}^{m_H} \text{Attention}(\mathbf{Q}\mathbf{W}_Q^{(h)}, \mathbf{K}\mathbf{W}_K^{(h)}, \mathbf{V}\mathbf{W}_V).\end{aligned} \quad (2)$$

$$GLU_w(\gamma) = \sigma(\mathbf{W}_{4,w} + \mathbf{b}_{4,w}) \odot (\mathbf{W}_{5,w} + \mathbf{b}_{5,w}) \quad (3)$$

$$GRN_w(\mathbf{a,c}) = LayerNorm(\mathbf{a} + GLU_w(\eta_1)) \quad (4)$$

$$\mathcal{L}(\Omega, \mathbf{W}) = \sum_{y_t \in \Omega}\sum_{q \in Q}\sum_{t=1}^{\tau_{max}} \frac{QL(y_t, \widehat{y}(q, t-\tau, \tau), q)}{M\tau_{max}}$$
$$QL(y, \hat{y}, q) = q(y-\hat{y})_+ + (1-q)(\hat{y}-y)_+ \quad (5)$$

We implemented the Multimodal Fusion Model as seen above in Figure 1, which obtained training data in 90 day windows, to create predictions in 15 day windows. The main model, the TFT Model, interacts with the Autoformer, ViT, and LLM models (though we were only able to achieve interaction with the Autoformer model, due to issues noted later in this section). The TFT Model sends a query into the Cross Attention Head, defined by equations 1 and 2, and continues to do that for all the succeeding models, obtaining only relevant information that can bolster predictions; after passing through the Cross Attention Heads, it passes through a gated linear unit, defined by equation 3 - this process can be skipped by having the TFT go directly through the Add and Norm Block, with a Layer Norm equation denoted by equation 4. Ideally, this Cross Attention will enrich the TFT output with each subsequent model. The gated linear unit will suppress any irrelevant information from the Cross Attention. The skip layer will allow the Fusion to train efficiently, preventing vanishing gradient and overfitting problems. Finally, once predictions are made, it is weighed against the Pinball Loss Function, which is defined by equation 5.We decide to have the fusion model to predict 3 quantile levels (10%, 50%, 90%) for interpretability and robustness.

We anticipated that most of our problems would stem from combining models to create our Multimodal model, as well as optimizing individual models and robustness. These anticipated problems, actually, became part of the problems that our team encountered. More specifically, due to computational constraints, we were not able to combine all the models as intended. The first thing we tried was implementing all models. However, we quickly ran into issues with the ViT and the LLM Text Encoder, which due to time, hardware, and tensor size constraints were not included. Focusing on the ViT, this model was restricted by Google Colab's CPU, 2 main methods were used: 1) Storing the images created locally and training with saved images, and 2) Creating the images dynamically and training with temporarily created images. For the Autoformer, the primary issue we faced was the gross overfitting of the model. The model's main issue (skipped over due to a lack of time) used a learning rate that was too high, and a very high hidden size dimension, and did not 'cool down' the teacher forcing. Although in hindsight this may have been something that contributed to the success of the fusion model, it is highly likely that improving the Autoformer may have lead to even stronger results.

Method 1 was not viable as saving the plotted images lead to the inability to edit the plotted features later. As mentioned earlier, more features were required to be present on the graph to create a more robust tensor to feed into the model; thus, Method 1 restricted the freedom of choosing features, reducing our flexibility in feature selection with this method. Furthermore, this was a time-consuming process - to plot the entire dataset while maintaining our fixed training window of 90 days, we would have to create 500,000 images, taking about 12 hours per 100,000 images when tested. This resulted storage and CPU timeout issues.

Method 2 was our method of choice - however, creating images dynamically meant increased time in loading batches to train the model. This resulted in training times of 12 hours per epoch. The LLM ran into similar issues described, hence both models were discarded from use within the larger model. It was determined that it was impossible to get substantial training done to optimize these models due to image creation and hardware limitations - time taken would encompass multiple days for the model to fully train, which in multiple iterations became our time constraint.

3. Experiments and Results

The main measurement of success for our Fusion model is its ability to outperform all of its sub-models. Our training process of the fusion model that contains only TFT and auto-formers took us about 2-3 hours on an RTX 2060 GPU. First, We trained TFT and Autoformer individually. Afterward, we used these 2 trained weights as the pre-trained weights for the according model with the Fusion model. Then, we trained the Fusion Model on the whole dataset.

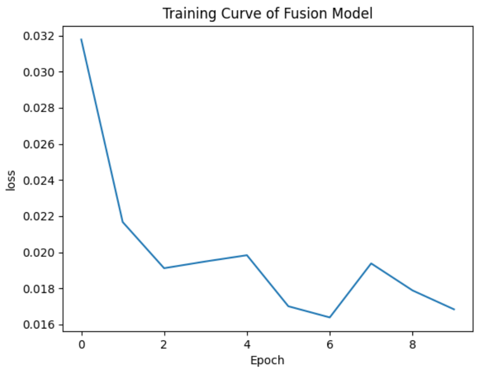
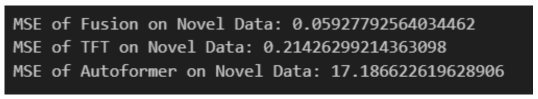


Figure 2. Fusion Model Training Curve

We achieved an MSE of 0.0115 on our test dataset. This value isn't very exciting and it gives some sign of over fitting. However, the most fascinating finding is the fusion model performance on novel data. Our training dataset is strictly from 2017 to 2022; the MSE of test dataset is 0.0115. We tested all 3 models on AMD from 12/11/2023 to 12/10/2024 and achieved the result below:



```
MSE of Fusion on Novel Data: 0.059277925640 34462
MSE of TFT on Novel Data: 0.21426299214363098
MSE of Autoformer on Novel Data: 17.186622619628906
```

Figure 3. AMD: MSE from 12/11/2023 to 12/10/2024

Based on the result above, we can clearly see that the Autoformer is over fitted and TFT also performed poorly on the novel dataset. The Fusion model, not only exceed at test dataset, but also outperforms both TFT and Autoformer by a huge magnitude.

Visually we can also come to other conclusions. The TFT and Autoformer models both predicted the price movement fairly well - however, it seemed to capture the

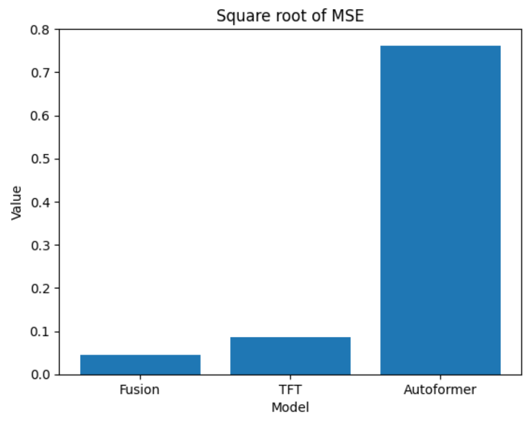


Figure 4. Bar Chart of all models for AMD: MSE from 12/11/2023 to 12/10/2024

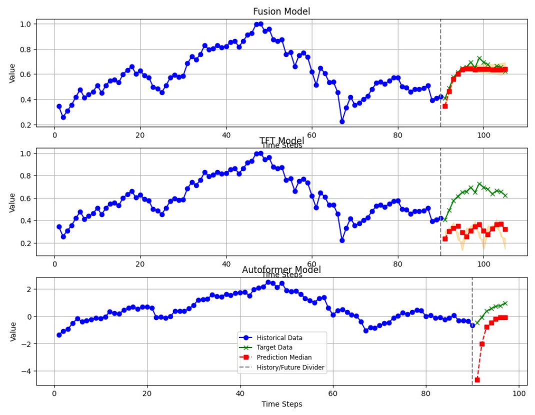


Figure 5. AMD stock price prediction (90 days window) from 12/11/2023 to 12/10/2024

volume that movement correctly. The autoformer, in that sense, performed better in capturing that volume, producing results that were closer to the general expected area, but were not as accurate at tracking that movement compared to the TFT Model. In this case, this is where the larger multimodal model outperforms both indivdual models - by taking optimal changes found in both composing models, it is able to more accurately reproduce the price movement in the correct general location.

However, it must be important to note that the accuracy, and reduced MSE, is only present within the first 7 days of prediction time - this is certainly due to the Autoformer model being trained on predicting only the first 7 days after the 90 day window. With lack of information from both models, as seeing that the TFT Model was also far off in the same time frame, the larger multimodal model was unable to generalize properly after the first 7 days. This created the flat line visible after the first 7 days of prediction in the figure above - the model could not create a pattern.

For hyperparameter tuning, we got these parameters:

hidden size = 128, dropout = 0.2, number of attention heads = 4, learning rate= 0.0001, momentum = 0.9, max grad norm = 1. We decided to have the same hidden size as the 2 baseline models for consistent information/gradient flow. For the optimizer, we used Adam to take advantage of the adaptive learning rate and momentum feature.

4. Individual Models

This section will delve into summaries for individual models, and results found from individual performance.

4.1. Temporal Fusion Transformer (TFT)

TFT model is the main model that the Fusion model used to enrich. This model was developed and designed by Google Cloud AI [4]. As a baseline model, TFT is designed to capture temporal dependencies and static covariates as feature input. In our version, static covariates information is obtained from Yfinance. There are: industry, sector, country, beta, marketCap, book value, dividend rate, dividend yield, five year Avg dividend yield, and debt to equity. The original implementation of TFT from google only used region as their static covariates information. Our implementation also uses a similar time-temporal encoding; we use day, month, day of the week, and week of the year. TFT encodes these time dependency features, static covariates, and stock data as input for the model. We followed the model architecture below and google's GitHub strictly as guidance for our implementation.

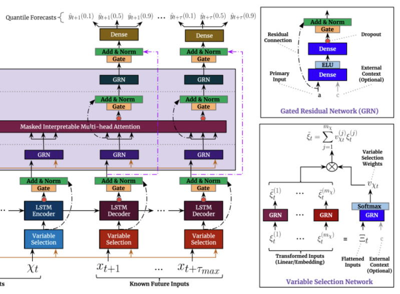


Figure 6. TFT Architecture Used

For our implementation, we tuned the hyper-parameter and obtained the best parameters to be: hidden size = 128, dropout = .2, number of attention head = 4, lr= 0.0001, momentum =.9, and max grad norm = 1. By itself, TFT did a decent job; it achieved an MSE of 0.07687681168317795 on the test dataset and it achieved an MSE Of 0.21689856052398682 on novel dataset (AMD stock data from 12/10/2023 to 12/10/2024

4.2. Autoformer

The Autoformer used was primarily based on the paper by Haixu Wu [6]. The architecture followed a struc-

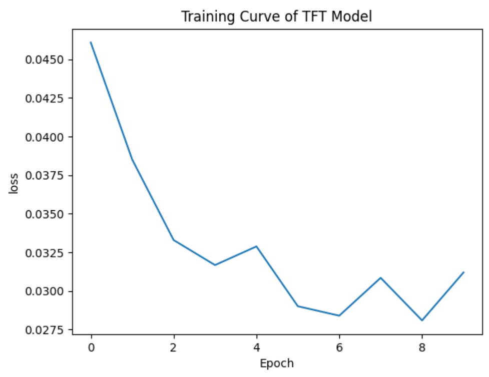


Figure 7. TFT Training Curve

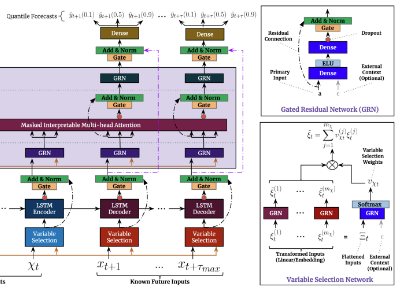


Figure 8. TFT Architecture Used

ture shown in Figure 4, but instead of using the autocorrelation attention, a standard multi-headed attention was used. In a brief overview, the model is fed in a window of time-ordered values and adds a decomposition layer that will allow us to split our series into seasonality and trend data. It is first used in the Encoder, and is sent to the Decoder, where the series will be processed in this way once again. The model, then is trained similarly to a standard transformer - it utilizes self-attention, adds prior seasonal data, and runs the data through a feed-forward network, which is then fed to This model is a counterbalance to the Temporal Fusion Transformer and is utilized to predict for a shorter time duration. In addition to this, the model was trained as label-agnostic - the specific intent behind this model was to be able to forecast different types of trends regardless of tickers or dates. The model also makes use of a concept known as teacher forcing, which provides target data to properly condition the model. Without tuning and making use of various tools, the model performs rather poorly, and harshly underfits. In order to combat this, we made use of gradient clipping, early stopping, and slightly tuned hyperparameters. For this specific model, our hyper parameters were as such: our hidden size = 512, learning_rate = 0.005, our heads for attention (n_heads) = 8, our feed forward network dimensions (ff_dim) = 512, number of total encoder/decoder layers (num_layers) = 2, kernel size for decomposition (kernel_size) = 5, and our dropout probability (dropout) = 0.05, and a learning rate scheduler with parameters: patience = 2, factor = 0.5. With these parameters, the model actually ended up overfitting, but this presents itself as a boon for when our models combine. For individual results, the MSE for the model during training was 0.006, and testing was 0.0041. However when testing on completely new data, the model's MSE was roughly 20.

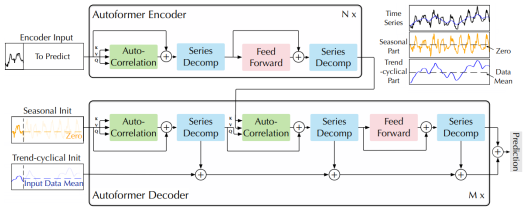


Figure 9. Autoformer Architecture

4.3. Large Language Model (LLM) Text Encoder

Data transformations were performed on Hanna Yukhymenko's Stock Tweets dataset to obtain a list of strings containing the concatenation of the stock ticker and the tweet contents. The strings were then converted into text embeddings using the SentenceTransformer embedding model [2] with nomic-embed-text-v1 [1], a pretrained LLM. Corresponding timestamps for each tweet that were set aside during the data transformation were then horizontally concatenated with the text embeddings to obtain text embeddings of tweets with preserved time features. This task was performed with several subsets of the data: random samplings of various percentages, and tweets of specific stocks. Unfortunately, the text embeddings were not able to be incorporated into the multimodal model due to time constraints and the large sizes of the tensors. .

4.4. Vision Transformer (ViT)

The ViT Model had very limited success, which led to its exclusion from the multimodal model. The model had two key issues:

1) Time spent training and dealing with image manipulation was the bottle-neck, with almost 14 hour training time **per** epoch. Pairing with the multimodal model would limit efficiency.
2) Poor performance, compared to TFT and other models in the larger multimodal model.

This approach was plagued by hardware limitations, which led to a minimum of 0.65 training loss which could not be improved due to time constraints. It is clear that this time bottle-neck was due to image creation - when monitoring image creation speeds. The majority of the ViT code was pulled from this repository, which was an implementation based on the paper, 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.' To improve performance, lines and modules were tweaked, as the original code produced subpar results, and a simple ViT model from Hugging Face was also used.

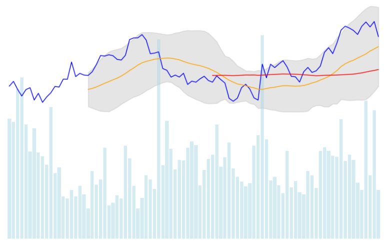


Figure 10. Image of Sample Stock with Feature Transformation

4.4.1 ViT Model Performance

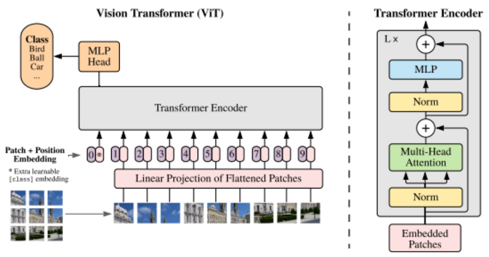


Figure 11. ViT Architecture Used

Our input consisted of turning the 64x64 image data into a flattened tensor with 4 patches of 16x16, which was fed into a customized ViT Model with an embedding dimension of 256, 8 heads, 10 layers, and 3 channels, paired with an Adam optimizer. The output was then 1 of 2 values - 1 or 0, to indicate the resulting rise or fall of the average price of the stock through a cumulative 15 days. These hyperparameters were chosen based on popular ViT architectures, as well as the need for a larger network to compensate for the model's lack of generalization. The image, once separated, was linearly projected, embedded, and then encoded through a transformer to produce results.

5. Future Work

Future work could see this larger Multimodal Model being improved with a variety of component models, that fo-

Student Name	Contributed Aspects	Details
Kai Alcayde	ViT, Paper Writing, and Project Analysis	Created ViT model and tailored data to fit for ViT. Supported research and writing for interpretation of larger multimodal model and results.
Luan Lam	TFT model, Fusion Model	Developed the TFT model and developed the Fusion Model. Developed the pipeline to collect and convert data to proper format for the fusion model.
Jake Yang	LLM Text Encoding, Paper Writing	Processed stock tweets, implemented text embedding model, and generated text embeddings. Contributed to paper and research, and performed proofreading.
Christopher Pak	Autoformer, Paper Writing, Data Visualization	Data Preprocessing, Training, and Testing for Autoformer model, also created code for visualizing forecasted prices.

Table 1. Contributions of team members.

cus on different feature spaces of the target data. For example, we were unable to implement ViT and the LLM Text Encoder into the prediction. However, their implementation could bolster the prediction and reduce MSE, as well as fix our generalization problem after the first 7 days - after all, our model is lacking the understanding to predict after this window, and a better understanding of that time frame and prediction window could greatly improve the robustness of our model.

6. Work Division

Shown below is the division of our work between our team. 1.

References

[1] nomic-embed-text-v1: A reproducible long context (8192) text embedder. https://huggingface.co/nomic-ai/nomic-embed-text-v1. Accessed: 2024-11-24. 5

[2] Sentencetransformers documentation. https://sbert.net/. Accessed: 2024-11-24. 5

[3] Zekun Li, Shiyang Li, and Xifeng Yan. Time series as images: Vision transformer for irregularly sampled time series, 2023. 1

[4] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020. 1, 4

[5] Paul Mooney. Stock market data (nasdaq, nyse, sp500). https://www.kaggle.com/datasets/paultimothymooney/stock-market-data. Accessed: 2024-11-24. 1

[6] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2022. 1, 4

[7] Hanna Yukhymenko. Stock tweets for sentiment analysis and prediction. https://www.kaggle.com/datasets/equinxx/stock-tweets-for-sentiment-analysis-and-prediction. Accessed: 2024-11-24. 1

[8] Chang Zong and Hang Zhou. Stock movement prediction with multimodal stable fusion via gated cross-attention mechanism, 2024. 1